AF $

IFW

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of

SHEN, HongHai

Appln. No.: 09/512,738

Confirmation No.: 5283

Filed: February 24, 2000

Docket No: A8642 / ST9-99-151

Group Art Unit: 2178

Examiner: Joshua D. Campbell

For: PROVIDING DYNAMIC WEB PAGES BY SEPARATING SCRIPTS AND HTML CODE

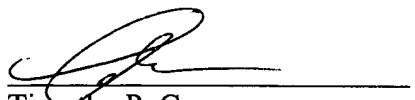
SUBMISSION OF APPEAL BRIEF

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Submitted herewith please find an *Appeal Brief*. The USPTO is directed and authorized to charge the statutory fee of $500.00 (and all required fees, except for the Issue Fee and the Publication Fee), to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account. A duplicate copy of this paper is attached.

Respectfully submitted,



Timothy P. Cremen
Registration No. 50,855

SUGHRUE MION, PLLC
Telephone: (202) 293-7060
Facsimile: (202) 293-7860

WASHINGTON OFFICE
23373
CUSTOMER NUMBER

Date: August 5, 2005

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of

SHEN, HongHai

Appln. No.: 09/512,738

Confirmation No.: 5283

Filed: February 24, 2000

Docket No: A8642 / ST9-99-151

Group Art Unit: 2178

Examiner: Joshua D. Campbell

For: PROVIDING DYNAMIC WEB PAGES BY SEPARATING SCRIPTS AND HTML CODE

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 41.37, Appellants submit the following:

Table of Contents

I. REAL PARTY IN INTEREST

The real party in interest is INTERNATIONAL BUSINESS MACHINES CORPORATION by virtue of an assignment executed by HongHai Shen and Yudong Sun (hereinafter "Appellants") on February 16, 2000.

II. RELATED APPEALS AND INTERFERENCES

To the best of the knowledge and belief of Appellants, the Assignee and the undersigned, there are no other appeals or interferences before the Board of Appeals and Interferences ("the Board") that will directly affect, or be affected by, the Board's decision in the present Appeal.

III. STATUS OF CLAIMS

Claims 1-54 are all the claims pending in the Application.

Claims 1-4, 7-14, 17-24 and 27-54 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Jamtgaard* (US 6,430,624 B1; hereinafter "*Jamtgaard*") in view of *Lipkin* (US 6,721,747; hereinafter "*Lipkin*").

Claims 5, 15 and 25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Jamtgaard* in view of *Lipkin* and *Maslov* (US 6,538,673; hereinafter "*Maslov*").

Claims 6, 16 and 26 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Jamtgaard* in view of *Lipkin* and *Tadakoro et al.* (US 6,463,352; hereinafter "*Tadakoro*").

IV. STATUS OF AMENDMENTS

A *Response Under 37 C.F.R. § 1.116* was filed on March 31, 2005, in response to the Final *Office Action* dated January 31, 2005. No changes were made to the claim set by way of the March 31, 2005 *Response*, and no other amendment or response was filed subsequent to the January 31, 2005 Final *Office Action*.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

To explain the invention for the Board's convenience, Appellants will first describe the relevant art (pages 1-6 of the Specification), and then the exemplary embodiments of the invention (pages 11-23 of the Specification). Portions of the claims that correspond to the features shown in the exemplary embodiments are also referenced during this discussion (portions of independent claims 1, 11 and 21 are provided in block quotes for easy identification). This discussion of the exemplary embodiments and the pending claims is provided for explanatory purposes only, and is not intended to limit the scope of the claims.

V(I). Relevant Art

Hypertext markup language ("HTML") is a common authoring language for creating Web pages. Until recently, most Web pages were static (*i.e.* the content was always the same). Thus, to produce customized content for different users, unique Web pages had to be created in advance for each user (App. pg. 2, lines 4-9).

Accordingly, techniques were developed to provide web pages that change each time they are viewed, based upon, for example, the user or current time. Such techniques included embedding server-side scripts in the Web pages, which execute on a Web server and generate HTML elements prior to the Web page being sent to a web browser (*e.g.*, Internet Explorer). Such a web page 2 with embedded scripts 4A-4C is shown in FIG. 1, which is reproduced to the right (App., pg. 2, lines 10-14; pg. 2, line 20 - pg. 3, line 2; pg. 4, lines 1-4).

In general, the embedded scripts 4A-4C are unintelligible to Web browsers and HTML editors. As such, if the Web page 2 of Figure 1 is displayed by a standard Web browser or HTML editor, the scripts will be ignored, and a displayed page 2 similar to that of Figure 2 will result (App., pg. 4, lines 5-8).

File: personalinfo.asp

```
<html>
<head>
<title>Personal Information Update</title>
</head>
<body>                                                          2
<table>
<form action="update.asp" method="post">
<tr>
<td>Name:</td>
<td>                                                            4A
<% Response.Write("<input name='name' value='"
    & rs("name") & "'>" ) %>
</td>
</tr>
<tr>
<td>Phone:</td>
<td>                                                            4B
<% Response.Write("<input name='phone' value='"
    & rs("phone") & "'>" ) %>
</td>
</tr>
<tr>
<td>E-mail:</td>
<td>                                                            4C
<% Response.Write("<input name='email' value='"
    & rs("email") & "'>" ) %>
</td>
</tr>
<td colspan="2" aling="center">
<input type="Submit" name="Submit" value="Submit">
</td>
</tr>
</form>
</table>
</body>
</html>
```

Fig. 1
(prior art)

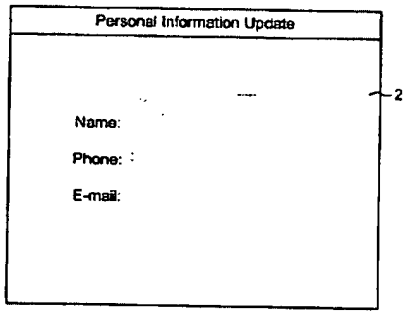


Fig. 2
(prior art)

To display data, the Web server handling a client request modifies the Web page 2 by replacing the scripts 4A-4C with the output of the script execution (*e.g.*, the Write() arguments). Typically, the output includes HTML elements 6A-6C, as illustrated in Figure 3. Thereafter, a "modified" Web page 8 may be sent to the requesting Web browser (App., pg. 4, lines 9-18). The modified Web page 8, as displayed by a Web browser, is shown in Figure 4. As a result of the above-described process, a single requested Web page 2 may produce customized output for different users (App., pg. 4, line 19 - pg. 5, line 2).

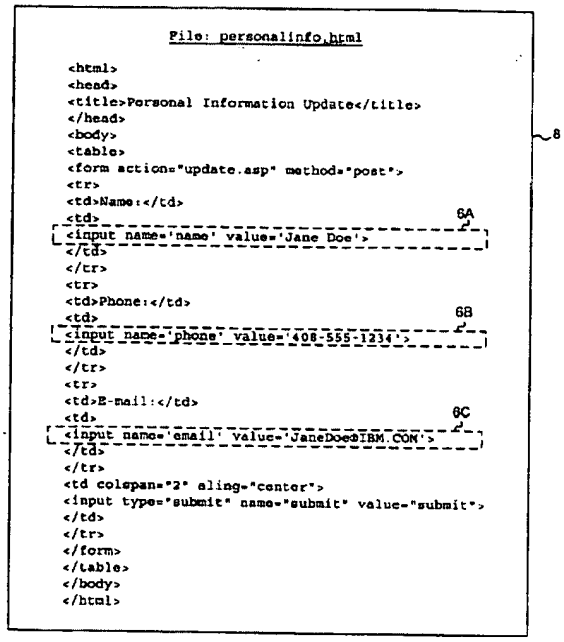


Fig. 3
(prior art)

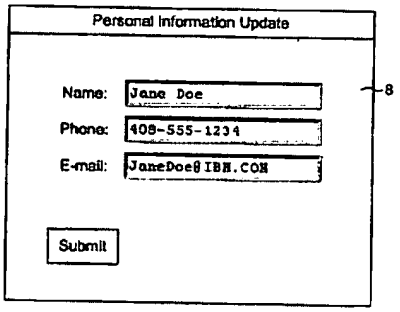


Fig. 4
(prior art)

However, embedding server-side scripts within Web pages has at least two major drawbacks. First, a Web page including such embedded scripts cannot be effectively edited with an interactive HTML editor, because some of the HTML elements of the page are only generated by the scripts at run time, and are thus unknown to the editor at design time. Second, Web documents including embedded scripts are often difficult to maintain and debug since the

scripts are typically scattered throughout a Web page at various locations (App., pg. 5, lines 3 - 21).

V(II). Exemplary Embodiments

Accordingly, to eliminate the above drawbacks, Appellants have developed a system that provides dynamic Web pages with separated scripts and HTML code so that the Web pages may be edited by an HTML editor.

An exemplary first embodiment of the invention is shown in Figures 6-15. Figure 6 shows a workstation 12 (with a web browser 52) and a web server 46. Web server 46 includes:

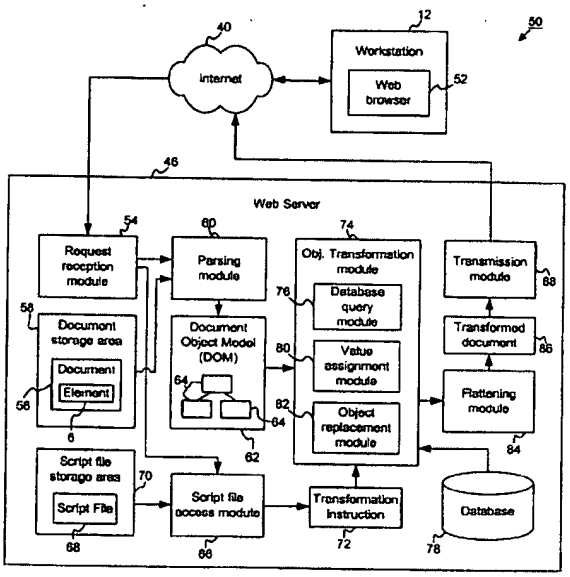


Fig. 6

(1) a request reception module 54 that receives from the web browser 52[1] a request for an HTML document 56 stored in document storage area 58;

(2) a parsing module 60, which retrieves and parses the requested document 56 to generate therefrom a corresponding Document Object Model (DOM) 62 with objects 64 (conventional web servers do not include a parsing module, since a document is normally parsed by a web browser);

[1] See dependent claims 4, 14 and 24.

(3) a script file access module 66, which retrieves a script file 68 from a script file storage area 70. The script file 68, corresponding to the requested document 56, contains transformation instructions, or "scripts," 72. The script file 68 may be entirely separate from, or simply in a separate portion of, document 56.[2] Unlike the conventional server-side scripts 4 of Figure 1, each transformation instruction 72 is directed to at least one object 64 of the DOM 62 and includes at least one transformation to be performed on that object 64;

(4) an object transformation module 74, which transforms one or more objects 64 of the DOM 62 in accordance with the transformation instructions 72 of a corresponding script file 68;

(5) a flattening module 84, which flattens the DOM 62 to generate therefrom a transformed document 86; and

[2] See dependent claims 6, 7, 16, 17, 26, 27 and 34-36.

(6) a transmission module 88, which sends the transformed document 86 to the workstation 12 and web browser 52 (App., pg. 14, line 4 - pg. 18, line 2).

As a matter of further description, a method 100 according to the invention is shown in Figure 7. In this method, at step 102, a request for document 56 is received by web server 46. Document 56 is an HTML

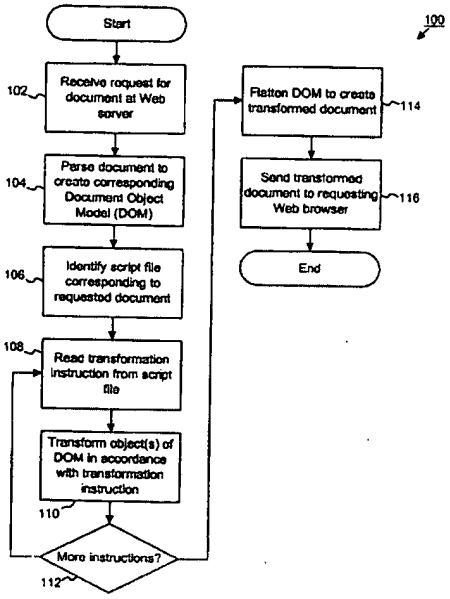


Fig. 7

[3] See dependent claims 31-33.

document configured as shown in FIG. 8, where instead of the script 4A of Web page 2 in FIG. 1, a regular HTML element 6D, *i.e.* < input name = 'name' value = ' ' >, may be used, where the "value" attribute may be left empty[3] (App., pg. 18, lines 3-16).

When displayed in a web browser (see FIG. 9), document 56 appears very similar to Figure 4, although the customized personal data shown in Figure 4 is not shown in Figure 9 (App., pg. 18, lines 17-18).

Thus, unlike the Web page 2 of Figure 1, document 56 may be effectively edited by an HTML editor, which is a great advantage to Web designers and HTML programmers, who need to edit a document that is as similar to the desired end product as possible (pg. 18, lines 19-21).

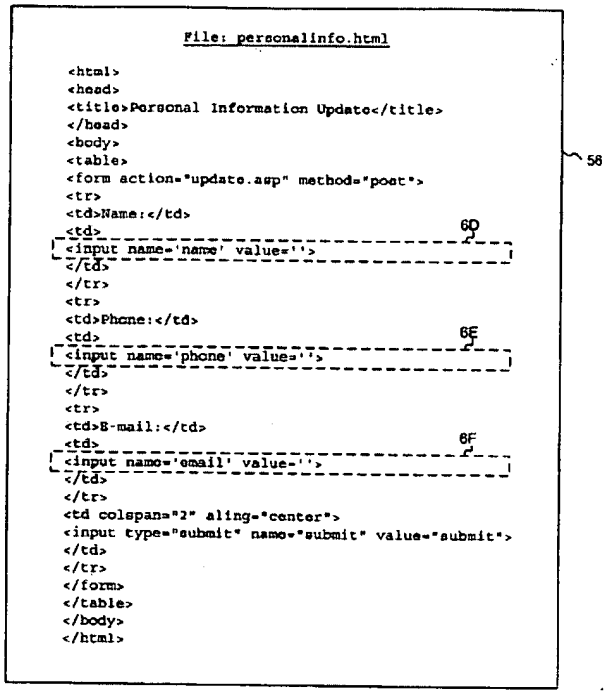


Fig. 8

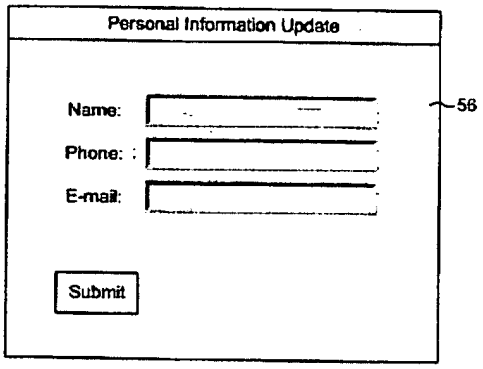


Fig. 9

The above corresponds to the following portions of independent claims 1, 11 and 21:

1. Within a document server, a computer-implemented method for processing a request for a document comprising at least one hypertext markup language (HTML) element, the method comprising:	*11.A system for processing a request for a document comprising at least one hypertext markup language (HTML) element, the system comprising:*	*21. An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by the processor to perform a computer-implemented method for processing a request for a document comprising at least one hypertext markup language (HTML) element, the method comprising:*

The next step shown in the method 100 of FIG. 7 is parsing 104 the document 56 to generate therefrom a corresponding DOM 62. As shown in Figure 10, a DOM 62 is a tree-like, hierarchical data structure including one or more objects 64 that represent the HTML elements 6 of the document 56 (App., pg. 19, lines 3-18).

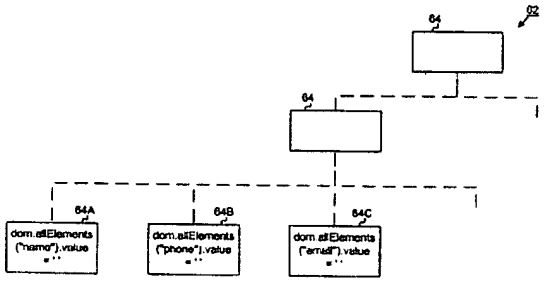


Fig. 10

The above corresponds to the following portions of independent claims 1, 11 and 21:

1. ... parsing the requested document to generate therefrom a corresponding document object model (DOM) including at least one object ...	*11. ... a parsing module configured to parse a requested document to generate therefrom a corresponding document object model (DOM) including at least one object ...*	*21. ... parsing the requested document to generate therefrom a corresponding document object model (DOM) including at least one object ...*

The next steps shown in the method 100 of FIG. 7 are identifying 106 and reading 108 a

script file 68 corresponding to the document 56. As noted briefly above, a document 56 and a corresponding script file 68[4]: (1) may be separate and have identical or similar names (*e.g.*, "personalinfo.html" and "personalinfo.scr"); or (2) may be separate portions of the same document.[5] An exemplary script file 68, with transformation instructions 72, is illustrated in FIG. 11 (App., pg. 19, line 9 - pg. 20, line 15).

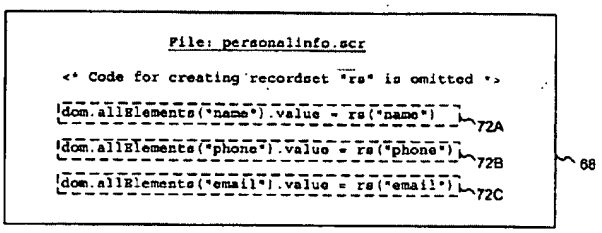


Fig. 11

The above corresponds to the following portions of independent claims 1, 11 and 21:

1. ... obtaining a transformation instruction directed to a first object of the DOM, the first object having a value ...	*11. ... an instruction obtaining module configured to obtain a transformation instruction directed to a first object of the DOM, the first object having a value ...*	*21. ... obtaining a transformation instruction directed to a first object of the DOM, the first object having a value...*

The next step shown in the method 100 of FIG. 7 is transforming 110 one or more objects 64 of the DOM 62 in accordance with the read transformation instruction 72.

[4] See dependent claims 2, 3 12, 13, 22 and 23.

[5] See dependent claims 6, 7, 16, 17, 26, 27, 34-36 and 37-39.

For example, if the first transformation instruction 72A is read, *i.e.* dom. allElements ("name") . value = rs ("name"), the method 100 may proceed to transform the object 64A of Figure 10 by querying a database 78 for a value, *i.e.* the user's name, and assigning the value to the object 64A.[6] One advantage of this method of transforming a DOM 62 rather than modifying a Web document 56, itself, as in conventional approaches, is that the DOM 62 is more easily transformed than HTML text (App., pg. 20, line 16 - pg. 21, line 9).

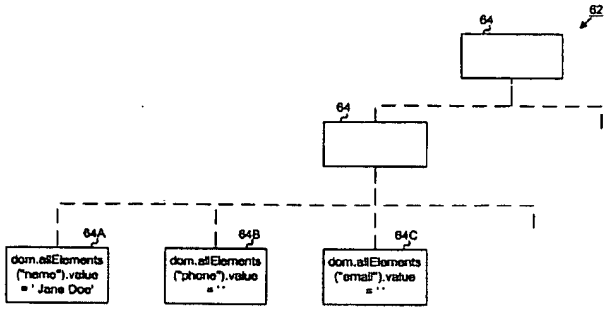


Fig. 12

The above corresponds to the following portions of independent claims 1, 11 and 21:

1. ... transforming the first object by changing the value thereof in accordance with the transformation instruction ...

11. ... an object transformation module configured to transform the first object by changing the value thereof in accordance with the transformation instruction ...

21. ... transforming the first object by changing the value thereof in accordance with the transformation instruction ...

The next step shown in the method 100 of FIG. 7, after determining 112 whether the script file 68 includes more transformation instructions 72, is flattening 114 the DOM 62 to create a transformed document 86.

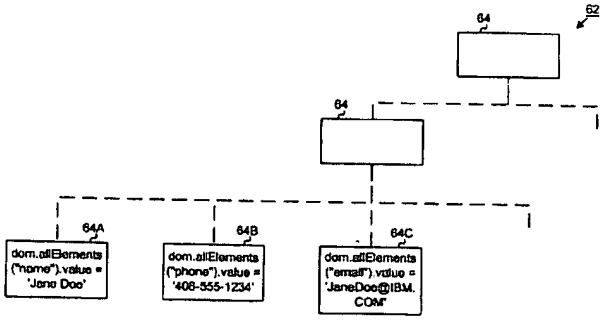


Fig. 13

[6] See dependent claims 9, 10, 19, 20, 29, 30, 42-44, 47-49 and 52-54.

[7] See dependent claims 40, 41, 45, 46, 50 and 51 - document 56 and 86 are both in HTML.

As previously noted, the flattening process involves converting the DOM 62 back into an HTML document 86.[7] Thus, any transformations to the DOM objects 64 will be reflected in the corresponding HTML elements 6 of the document 86. For example, Figure 13 illustrates the DOM 62 after execution of the three transformations instructions 72A-C, and Figure 14 illustrates the transformed document 86 after flattening 114, which may then be sent 116 to the requesting Web browser 52 and displayed, as illustrated in Figure 15 (App., pg. 21, line 10 - pg. 22, line 2).

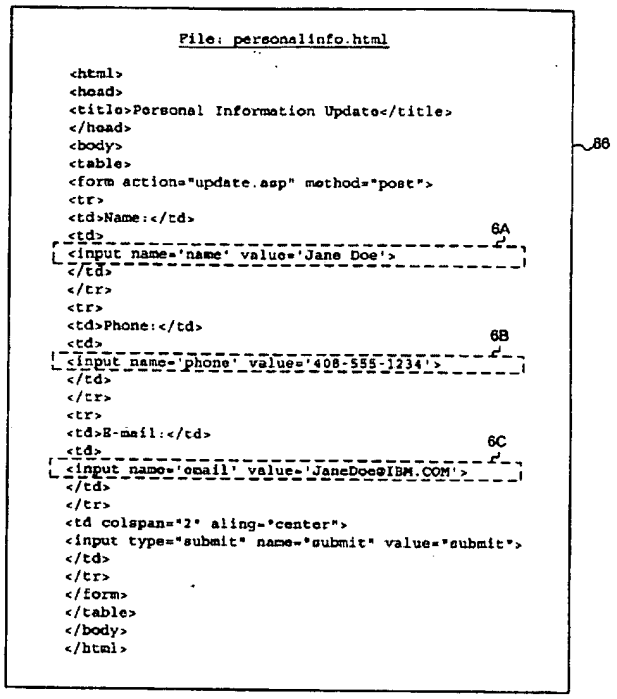


Fig. 14

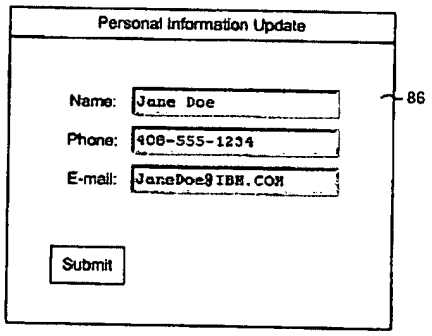


Fig. 15

The above corresponds to the following portions of independent claims 1, 11 and 21:

1. ... flattening the DOM to generate therefrom a corresponding transformed document

11. ... a flattening module configured to flatten the DOM to generate therefrom a corresponding transformed document.

21. ... flattening the DOM to generate therefrom a corresponding transformed document.

Thus, the transformed document 86 of the invention is similar to the conventional modified document 8 of Figure 3. However, the transformed document 86 of the invention does

not rely on embedded scripts 4, and the transformation instructions 72 are not "place holders" for HTML elements 6 to be inserted later by a Web server 46. As such, all of the transformations instructions 72 can be located together, even within a separate script file 68 (App., pg. 22, lines 3-13).

Thus, a document 56 in accordance with the present invention may be effectively edited by an HTML editor, since all of the HTML elements may be included within the document 56 at the time of design. Any transformations, such as assignments of values and the like, may be accomplished by transforming the DOM 62, and no embedded scripts 4 are necessary (App., pg. 22, lines 14-18).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether or not:

(1) claims 1-4, 7-14, 17-24 and 27-54 are unpatentable over *Jamtgaard* in view of *Lipkin*;

(2) claims 5, 15 and 25 are unpatentable over *Jamtgaard* in view of *Lipkin* and *Maslov*;

and

(3) claims 6, 16 and 26 are unpatentable over *Jamtgaard* in view of *Lipkin* and *Tadakoro*.

VII. ARGUMENT

A. Independent Claims 1, 11 and 21 - The Examiner's Position

As noted above, the Examiner has taken the position that claims 1-4, 7-14, 17-24 and 27-54 are unpatentable over *Jamtgaard* in view of *Lipkin*. More specifically, the Examiner alleges that *Jamtgaard* discloses (O.A., pg. 3) that:

> [1] an HTML file is requested via the internet (column 2, lines 40-59 of Jamtgaard), "... processing a request for a document comprising at least one hypertext markup language (HTML) element" [;]
>
> [2] the HTML files [are] parsed and translated into a document object model tree (column 9, lines 48-63 of Jamtgaard), "... parsing the requested document to generate therefrom a corresponding document object model (DOM) including at least one object" [;]
>
> [3] transformation instructions are obtained that correspond to a document URL that dictate how to convert the HTML file (beginning with the first object) into relational markup language, and the conversion (transformation) is performed (column 10, line 20 - column 11, line 12 of Jamtgaard), "... obtaining a transformation instruction directed to a first object of the DOM" and "...transforming the first object in accordance with the transformation instruction" [; and]
>
> [4] the converted document is then output to the requesting device by transforming portions of the DOM tree back into complete documents (flattening) called cards (column 14, lines 4-21 of Jamtgaard), "...flattening the DOM to generate therefrom a corresponding transformed document".

The Examiner concedes that *Jamtgaard* "does not directly disclose a method in which the transformation of an object consists of changing the value of that object" (O.A., pgs 3-4). Applicant agrees that *Jamtgaard* fails to teach or suggest such features, for at least the reasons discussed in the April 29, 2004 *Amendment*.

Nevertheless, the Examiner alleges that such features are disclosed by col. 69, line 1 - col. 71 line 15 of *Lipkin*, alleging that it "discloses a method in which a transformation of a DOM object consists of changing the underlying value of that object," citing column 69, line 1 -

column 71, line 15 (O.A., pgs 3-4). The Examiner further explains his interpretation in the April 6, 2005 *Advisory Action*, alleging that *Lipkin* discloses "that the value of the xsp node may be replaced based on the ... XSL (transformation instruction) in a way that is transparent to user," citing column 69, lines 1-24 (A.A., attachment page).

Thus, the Examiner alleges that:

> [i]t would have been obvious to one of ordinary skill in the art at the time the invention was made to have combined the methods of Jamtgaard with method [sic] of Lipkin because it would have allowed for a transformation to occur that was transparent to the user.

Appellants respectfully disagree.

B. The Applied References

Jamtgaard discloses a content delivery system 10 with a translation server 12 that takes information from a content provider's website in, *e.g.*, HTML form, and translates and redelivers it in a customized form for an end user (col. 4, lines 58-66).

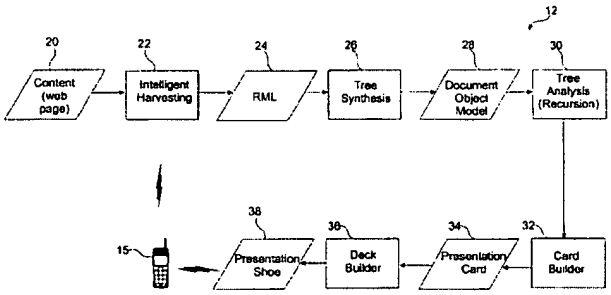


FIG 3

More specifically, as shown in FIG. 3, intelligent harvester 22 receives the provider's content and generates a relational data structure, RML 24, which is then transformed into a DOM 28 by tree synthesizer 26 (col. 5, lines 54-65). The tree analysis function 30 receives the DOM and generates pages for a variety of different screen sizes (col. 6, lines 11-14). Card builder 32 and deck builder 35 then build the content for sending to the device 15 (col. 6, lines 20-31).

Thus, when non-PC devices 15 request a web page, the web page information can be translated by translation server 12 into a data format appropriate for and recognizable to the

destination device 15. However, PCs do not need such translation server 12, because they are able to display the web page in standard HTML (col. 7, lines 22-30).

Accordingly, Appellants respectfully submit that *Jamtgaard* is directed only to reformatting existing web pages to change their format from a first format (*e.g.*, HTML) to another format so that they may be viewed on various non-PC (*e.g.*, PDA) devices.

Lipkin generally discloses a management system for an information server (see FIG. 4) for generating (*i.e.*, creating from scratch) web pages. Interface server 417 (also called WDK server) is provided to manage dynamic generation of content and to manipulate various kinds of display style sheets to display data on various devices such as web clients 401, laptop 407 and PDA 411 (col. 10, lines 40-64).

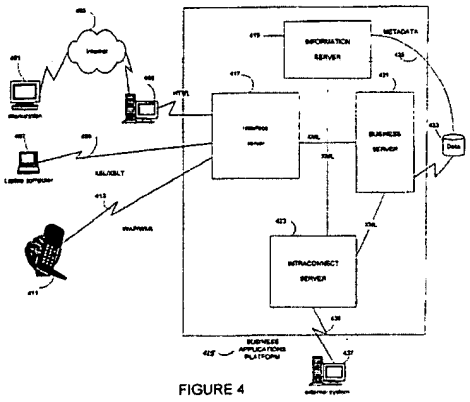


FIGURE 4

A WDK server 800 is further shown in FIG. 8A, and is disclosed as enabling interaction with users regardless of the user's specific hardware, software, and location (col. 48, lines 25-30).

In *Lipkin's* specific embodiment (see the description starting in col. 54, line 34), when a client sends a request to web content server 800, content production and presentation is achieved by following a

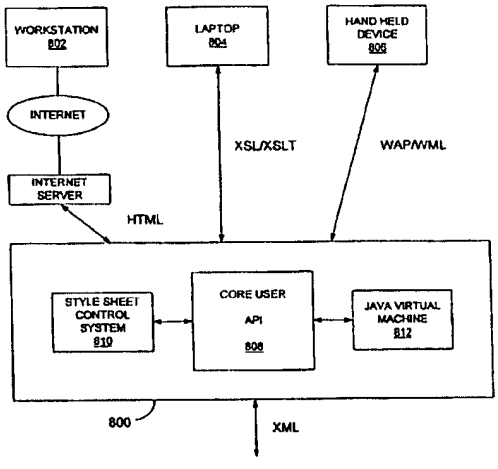


FIG. 8A

Model-View-Widget (MVW) paradigm, where: (1) a model page is provided to produce the web page contents; and (2) various user interaction components (called "widgets" in *Lipkin*) and view transformations provide the look and layout of the product displayed to the client (col. 54, lines 48-58).

More specifically, a control file is used by the web server to create a web page by: (1) identifying model, view and widget files; (2) creating a DOM representation of the XML model file; (3) updating the DOM to insert specific instructions for locations of XSL style sheets and XSLT processing instructions; (4) updating hyperlinks in the model file; and (5) returning the DOM so that it may be processed by the web server 800 (col. 56, lines 40-62). The steps of updating the DOM and hyperlinks consist of inserting various command lines and hyperlinks into the model page to customize the page that will be displayed to the client (col. 57, line 18 - col. 58, line 52; *compare* bolded portions of the table in cols. 61-62 to the table in cols. 59-60).

Further, *Lipkin* discloses that the XML model pages that are used to form the DOM are XSP pages that contain a mix of static content and content generated by using XSP tags, and consists of a head, form and widget portion. The form portion contains all data generated by the page (col. 66, lines 10-41). At the end of the processing of the XML model page, the widgets (*i.e.*, an input text field) and content (*i.e.*, a data value) have to be combined. This is accomplished by XSP attachTo tags (col. 69, lines 1-25).

Thus, Appellants respectfully submit that *Lipkin* is primarily directed to creating viewable pages from an XSP model page to display that viewable page on various devices. Hence, *Lipkin* is quite different from *Jamtgaard*, which simply reformats an existing HTML page to display it on non-PC devices.

C. Appellants' Traversal of the Rejections of Independent Claims 1, 11 And 21

Appellants respectfully disagree with the rejections of record, at least because: (1) *Lipkin* fails to teach or suggest the features that the Examiner has conceded are missing from *Jamtgaard*; and (2) one of skill would not have been motivated to modify *Jamtgaard* in view of *Lipkin*, at least in the manner alleged by the Examiner.

C(1). Lipkin Fails To Teach Or Suggest "Transforming The First Object By Changing The Value Thereof In Accordance With The Transformation Instruction"

Appellant respectfully submits that *Lipkin* fails to teach or suggest the conceded deficiencies of *Jamtgaard* - claim 1's recitation of "transforming the first object by changing the value thereof in accordance with the transformation instruction."

Specifically, Appellants respectfully submit that *Lipkin* discloses only two general types of operations performed on XSP model pages.

First, *Lipkin* discloses inserting new command lines containing processing instructions (col. 57, lines 7-40) and new links (col. 58, lines 19-58) to create the contents of the viewable page that is to be displayed. However, this addition of command lines does not affect the existing command lines of the XSP model page, as shown in the comparative tables in cols. 59-62 of *Lipkin*. Such insertion of new commands cannot reasonably be read as "changing" the value of a first object of a DOM, as recited in the independent claims. Further, Appellants respectfully submit that there is no specific teaching or suggestion of just how these command lines would be inserted with respect to the DOM disclosed in *Lipkin*, or how such an insertion even affects the DOM structure.

Second, *Lipkin* discloses the formatting of the XSP model page for a particular display type. This is accomplished by applying: (1) the "widgets" described above; and (2) an XSLT stylesheet, to the page.

The "widgets" of *Lipkin* are "input fields, buttons, hyperlinks, etc" (col. 66, lines 36-37), while the model element of the XSP page contains all data, or content, generated by the page (col. 66, lines 40-42). The widgets and content are combined at the end of processing so that the content can be displayed with the widgets (col. 66, lines 50-55). This is accomplished by attaching particular widgets to the content (col. 66, lines 57-58) by inserting the widgets therein, as shown in the tables in columns 67 and 68.

However, it is important to note that *Lipkin* fails to teach or suggest that this attachment of widgets to the page content changes the content of the page in any way. Rather, attaching these widgets only modifies how the content is displayed, not any underlying value. In fact, the Examiner has not cited a single portion of *Lipkin* that discloses modifying the underlying value of a DOM object, and *Lipkin* is no more relevant to the instant Application than *Jamtgaard.*

C(2). One Of Skill Would Not Have Been Motivated to Modify Jamtgaard in View of Lipkin

Appellant respectfully submits that even if *Lipkin* could be read as somehow disclosing the underlying modification of data (as the Examiner alleges), one of skill would still not have been motivated to modify *Jamtgaard* as the Examiner alleges.

Specifically, it has long been held that the Examiner must "show reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for a combination in the manner claimed." *In re Rouffet*, 47 USPQ2d 1453 (Fed.Cir. 1998). The mere fact that

references can be "combined or modified does not render the resultant combination [or modification] obvious unless the prior art also suggests the desirability of the combination [or modification]." *In re Mills*, 916 F.2d 680 (Fed.Cir. 1990); MPEP §2143.01.

As an initial matter, Appellants respectfully submit that the Examiner's proffered motivation to modify *Jamtgaard* in view of *Lipkin* - "to allow for a transformation to occur that is transparent to the user" - does not provide the requisite motivation discussed above. Specifically, *Jamtgaard* already provides an "automatic" reformatting of content that is transparent to the user (the reformatting of an existing page to be displayed on the user's device). Thus, *Jamtgaard* would not need to be modified to provide a feature that already exists.

Further, no other motivation to combine these references is immediately apparent from even a general reading thereof, as the basic differences between *Jamtgaard* and *Lipkin* are quite apparent. *Jamtgaard* is directed to reformatting an existing document so that it may be displayed on a particular client, while *Lipkin* is directed to creating and formatting a document from scratch so that it may be viewed on a particular client. Thus, while the method of *Jamtgaard* is tailored to operate on an existing mark-up language page (*e.g.*, HTML), *Lipkin* is tailored to operate on a specific XSP model page, upon which command lines are inserted, and a style sheet is applied to form a mark-up language page. Accordingly, even in the most general sense, *Jamtgaard* and *Lipkin* are: (1) directed to operate in two distinctly different roles (reformatting vs. creation); and (2) use as a starting point two distinctly different document types (an existing document vs. an XSP model).

Even more specifically, Appellants respectfully submit that *Jamtgaard* is directed to providing the same underlying data in different formats to be viewable on different devices - *i.e.*,

"a single piece of content that is re-formatted automatically for the different information appliances" (*Jamtgaard* Abstract). Thus, the Examiner's proffered modification of *Jamtgaard* to somehow also change the underlying data would be contrary to this stated purpose, as the alleged modification would result in a system that would no longer provide the same content in different formats, but would somehow provide different content in different formats.

Further, *Lipkin* is directed to the actual creation of a viewable page from scratch via an XSP model page, not the modification of an existing page. As part of this process, *Lipkin* applies a style sheet so that the XSP model page may be transformed into whatever format is required for the client device (as discussed above). Accordingly, the *Lipkin* process creates and formats a web page for viewing on a particular device, and therefore would have no need for a reformatting step such as is disclosed in *Jamtgaard*, at least as applied as proffered by the Examiner.

Still further, the *Lipkin* system is dependent upon the provision of an XSP model page as a starting point. This model page, as discussed above, then has command lines inserted therein, and a style sheet applied thereto, to form a mark-up language page for viewing by a client. However, such an XSP model page is not even considered by *Jamtgaard*. Rather, *Jamtgaard* starts with a mark-up language page, and then reformats it. Thus, *Lipkin* and *Jamtgaard* cannot reasonably be considered as alternative or consecutive processes. Rather, they can only reasonably be considered mutually exclusive, successive processes.

Thus, one of skill would not have been motivated to modify *Jamtgaard's* re-formatting system with a the page creation system of *Lipkin*.

Accordingly, Appellants respectfully submit that independent claims 1, 11 and 21 are patentable over the applied references.

C(3). Dependent Claims 2-4, 12-14 and 22-24

Appellants respectfully submit that dependent claims 2-4, 12-14 and 22-24 are allowable at least by virtue of their dependency from independent claims 1, 11 and 21, respectively.

C(4). Dependent Claims 5, 15 and 25

The Examiner concedes that *Jamtgaard* and *Lipkin* are silent regarding the features of claims 5, 15 and 25. However, the Examiner alleges that *Maslov* discloses:

> a method in which a user requests a script file to start the transformation of a document using a DOM tree and based on that script file the content source documents references by that script file are loaded (column 6, lines 1-13).

The Examiner further alleges that it would have been obvious to modify *Jamtgaard* with the method of *Maslov*:

> Because it would have allowed a user to reference more than one source document with one script file and have all of them loaded automatically and all necessary transformations preformed with only the request of one document.

Appellants respectfully disagree, and submit that the proffered combination fails to teach or suggest "receiving a request for a script file from a client program; and identifying a document within the document server corresponding to the requested script file," as recited in claims 5 and 25, or "a request reception module configured to receive a request for a script file from a client program and to identify a document corresponding to the requested script file," as recited in claim 15.

Specifically, Appellants respectfully submit that *Maslov* is directed (see FIG. 1) to obtaining fragments 15, 25 of source documents 10, 20 from the internet, and then displaying

them in a target window 30. Thus, *Maslov's* script is directed to identifying documents from the internet, not documents on the recited document server, as recited in these claims.

Additionally, Appellants respectfully submit that *Maslov* fails to teach or suggest the features missing from *Jamtgaard* and *Lipkin* discussed above.

C(5). Dependent Claims 6, 16 and 26

The Examiner concedes that *Jamtgaard* and *Lipkin* are silent regarding the features of claims 6, 16 and 26. However, the Examiner alleges that *Tadakoro* discloses "a method in which scripts can be separate from a file or embedded in an HTML file and function the same either way (column 12, lines 11-63)."

Appellants respectfully disagree, and submit that the proffered combination of *Jamtgaard*, *Lipkin* and *Tadakoro* fails to teach or suggest that "the script file is included within a separate portion of the document," as recited in claims 6, 16 and 26.

Specifically, the portion of *Tadakoro* cited by the Examiner refers only to script-embedded HTML pages, such as are shown in the instant Application's prior art Figure 1. Such script-embedded HTML pages fail to teach or suggest a script file that is in a separate portion of the HTML document. Rather, the script-embedded HTML pages provide script lines amongst the HTML tags, as shown in Figure 1 of the Application.

Additionally, Appellants respectfully submit that *Tadakoro* is silent regarding the features missing from *Jamtgaard* and *Lipkin* discussed above with respect to the independent claims.

C(6). Dependent Claims 7-9, 17-19, and 27-29

Appellants respectfully submit that dependent claims 7, 8, 17, 18, 27 and 28 are allowable at least by virtue of their dependency from independent claims 1, 11 and 21, respectively.

C(7). Dependent Claims 10, 20 and 30

The Examiner alleges that *Jamtgaard* discloses all of the features of claims 10, 20 and 30, specifically alleging that that *Jamtgaard* discloses "replacing the HTML document with the RML document (column 10, line 20 - column 11, line 12)."

Appellants respectfully disagree, and submit that the cited portion of *Jamtgaard* fails to teach or suggest the replacement of "a first object of the DOM with a different second object," as recited in claims 10, 20 and 30.

Specifically, as discussed in detail above, *Jamtgaard* discloses reformatting the HTML document to be an RML document by reformatting various objects. *Jamtgaard* does not teach or suggest the replacement of any particular DOM objects with other objects.

Appellants also respectfully submit that *Lipkin* is similarly silent regarding the replacement of particular objects, as it is only directed to either insertion of particular command lines, or reformatting existing elements (as discussed above).

C(8). Dependent Claims 34-36

Appellants respectfully submit that dependent claims 734-36 are allowable at least by virtue of their dependency from independent claims 1, 11 and 21, respectively.

C(9). Dependent Claims 31-33 and 37-39

The Examiner alleges that *Jamtgaard* discloses how to convert an HTML file into RML, citing col. 1, line 20 - col. 11, line 12, and that "transformation instructions are found in XSL files on the server that are associated with the URL of the requested document ... which includes the first object (column 6, lines 11-53)."

Appellants respectfully disagree, and submit the proffered combination fails to teach or suggest that "the first object is an HTML file," as recited in dependent claims 31-33, and that "the first object is an HTML file; the transformation instruction is read from a script file located separately from the HTML file; and the HTML file and the script file contain information to indicate their correspondence to each other," as recited in claims 37-39.

Specifically, the Examiner has rejected independent claim 1 as being unpatentable over a combination of *Jamtgaard* in view of *Lipkin*. In order to utilize the disclosure of *Lipkin* as the Examiner proffers, the resultant combination must use an XSP model sheet (or its equivalent) as the first object, not the HMTL page of *Jamtgaard*. Otherwise, the disclosed system of *Lipkin*, which is specifically relied upon by the Examiner, would not function.

Thus, while *Jamtgaard* by itself discloses an HTML page as its first object, the Examiner cannot cite this portion of *Jamtgaard*, as, even in the most general combination of the applied references, it must be changed to incorporate the modifications espoused by *Lipkin*.

In sum, it is the proffered combination that must be read relative to the pending claims.

C(10). Dependent Claims 41, 45 and 50

The Examiner concedes that *Jamtgaard* does not disclose a method in which the transformed document and the original document are in the same format. However, the

Examiner alleges that *Lipkin* discloses that "the transformation only occurs on the underlying values of the DOM objects, thus allowing the original document to remain in its format after the transformation, which as disclosed by Lipkin can be HTML, XML, and other languages," citing col. 49, line 36 - col. 50, line 59.

Appellants respectfully disagree, and submit the proffered combination fails to teach or suggest: (1) that "the document and the corresponding transformed document are in the same format," as recited in claims 40, 45 and 50; and (2) that "the same format is HTML," as recited in claims 41, 45 and 50.

Specifically, *Jamtgaard* discloses a system for transforming an HTML page to a page of another format for display on, for example, a portable device. Thus, the entire purpose of *Jamtgaard* is to change the format of the document.

Lipkin discloses forming a DOM from an XSP model page, adding command lines and modifying the format of the XSP model page, and then processing the DOM to provide a document in a format understood by a browser, such as HTML (col. 69, lines 55-65). There is no teaching or suggestion that, after processing the DOM, the document remains in an XSP format. Further, there is no teaching or suggestion that the XSP model page can, in any embodiment, be provided in HTML. Thus, Appellants respectfully submit that *Lipkin* also cannot be reasonably read as teaching or suggesting that a first document and transformed document can be in the same format.

C(11). Dependent Claims 42, 47 and 52

The Examiner concedes that *Jamtgaard* fails to teach or suggest the features recited in claims 42, 47 and 52. However, the Examiner alleges that *Lipkin* discloses that "data presented

to the user may be based on a login or user preferences, thus providing different information to different users (column 84, line 23 - column 85, line 19).

Appellants respectfully disagree, and submit the proffered combination fails to teach or suggest that "the value is changed in accordance with different users," as recited in claims 42, 47 and 52.

Specifically, the cited portion of *Lipkin* indicates that various data might be presented to different users on the final HTML page (*e.g.*, objects A, B and D might be presented to user 1, while objects A, C and D might be presented to user 2). However, the cited portion does not teach or suggest that the values of the objects themselves are any different, or that the values of those objects might be changed in accordance with a particular user.

C(12). Dependent Claims 43, 48 and 53

The Examiner concedes that *Jamtgaard* fails to teach or suggest these features. However, the Examiner alleges that *Lipkin* discloses that "objects may be empty until a request to view them is made, at which point the correct values, which can be variables, are then placed into the object (column 69, lines 1 - column 71, line 15).

Appellants respectfully disagree, and submit that the proffered combination fails to teach or suggest that "the value is a variable," as recited in claims 43, 48 and 53, or that "the value of the first object is empty before the first object is transformed," as recited in claims 44, 49 and 54.

Specifically, the portion of *Lipkin* cited by the Examiner is directed to the application of a XSLT stylesheet to the XSP model page. This application of the XSLT stylesheet only adjusts the format of the page, not the underlying data. Thus, the Examiner's rejection is unsupported.

Further, as noted above, *Lipkin* only discloses adding command lines and modifying the format of the XSP model page, not changing the value of any particular object.

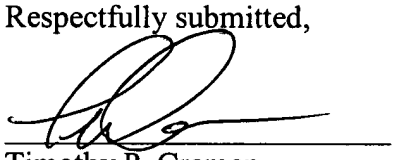
VIII. CONCLUSION

In view of the foregoing differences between appealed claims 1-54 and the cited references, Appellants respectfully submit that appealed claims 1-54 are patentable over the applied references.

Unless a check is submitted herewith for the fee required under 37 C.F.R. §41.37(a) and 1.17(c), please charge said fee to Deposit Account No. 19-4880.

The USPTO is directed and authorized to charge all required fees, except for the Issue Fee and the Publication Fee, to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account.

Respectfully submitted,



Timothy P. Cremen
Registration No. 50,855

SUGHRUE MION, PLLC
Telephone: (202) 293-7060
Facsimile: (202) 293-7860

WASHINGTON OFFICE
23373
CUSTOMER NUMBER

Date: DRAFT

CLAIMS APPENDIX

CLAIMS 1-54 ON APPEAL:

1. (Previously Presented) Within a document server, a computer-implemented method for processing a request for a document comprising at least one hypertext markup language (HTML) element, the method comprising:

parsing the requested document to generate therefrom a corresponding document object model (DOM) including at least one object;

obtaining a transformation instruction directed to a first object of the DOM, the first object having a value;

transforming the first object by changing the value thereof in accordance with the transformation instruction; and

flattening the DOM to generate therefrom a corresponding transformed document.

2. (Original) The method of claim 1, wherein the obtaining step comprises:

reading a transformation instruction from a script file corresponding to the requested document.

3. (Original) The method of claim 2, further comprising:

receiving a request for a document from a client program; and

identifying a script file within the document server corresponding to the requested document.

4. (Original) The method of claim 3, wherein the client program comprises a Web browser.

5. (Original) The method of claim 2, further comprising:

receiving a request for a script file from a client program; and

identifying a document within the document server corresponding to the requested script file.

6. (Original) The method of claim 2, wherein the script file is included within a separate portion of the document.

7. (Original) The method of claim 2, wherein the script file and the document comprise logically separate data files.

8. (Original) The method of claim 1, further comprising:

transmitting the transformed document to a client program.

9. (Previously Presented) The method of claim 1, wherein the transforming step comprises:

retrieving a database value from a database; and

assigning the database value to an object of the DOM.

10. (Original) The method of claim 1, wherein the transforming step comprises:

replacing a first object of the DOM with a different second object.

11. (Previously Presented) A system for processing a request for a document comprising at least one hypertext markup language (HTML) element, the system comprising:

a parsing module configured to parse a requested document to generate therefrom a corresponding document object model (DOM) including at least one object;

an instruction obtaining module configured to obtain a transformation instruction directed to a first object of the DOM, the first object having a value;

an object transformation module configured to transform the first object by changing the value thereof in accordance with the transformation instruction; and

a flattening module configured to flatten the DOM to generate therefrom a corresponding transformed document.

12. (Original) The system of claim 11, wherein the instruction module comprises:

a script file access module configured to read a transformation instruction from a script file corresponding to the requested document.

13. (Original) The system of claim 12, further comprising:

a request reception module configured to receive a request for a document from a client program and identify a script file corresponding to the requested document.

14. (Original) The system of claim 13, wherein the client program comprises a Web browser.

15. (Previously Presented) The system of claim 12, further comprising:

a request reception module configured to receive a request for a script file from a client program and to identify a document corresponding to the requested script file.

16. (Original) The system of claim 12, wherein the script file is included within a separate portion of the document.

17. (Original) The system of claim 12, wherein the script file and the document comprise logically separate data files.

18. (Original) The system of claim 11, further comprising:

a transmission module configured to transmit the transformed document to a client program.

19. (Previously Presented) The system of claim 11, wherein the object transformation module comprises:

a database query module configured to retrieve a database value from a database; and

a value assignment module configured to assign the database value to an object of the DOM.

20. (Original) The system of claim 11, wherein the object transformation module comprises:

an element replacement module configured to replace a first object of the DOM with a different second object.

21. (Currently Amended) An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by the processor to perform a computer-implemented method for processing a request for a document comprising at least one hypertext markup language (HTML) element, the method comprising:

parsing the requested document to generate therefrom a corresponding document object model (DOM) including at least one object;

obtaining a transformation instruction directed to a first object of the DOM, the first object having a value;

transforming the first object by changing the value thereof in accordance with the transformation instruction; and

flattening the DOM to generate therefrom a corresponding transformed document.

22. (Original) The article of manufacture of claim 21, wherein the obtaining step comprises:

reading a transformation instruction from a script file corresponding to the requested document.

23. (Original) The article of manufacture of claim 22, the method further comprising:

receiving a request for a document from a client program; and

identifying a script file corresponding to the requested document.

24. (Original) The article of manufacture of claim 23, wherein the client program comprises a Web browser.

25. (Original) The article of manufacture of claim 22, the method further comprising:

receiving a request for a script file from a client program; and

identifying a document corresponding to the requested script file.

26. (Original) The article of manufacture of claim 22, wherein the script file is included within a separate portion of the document.

27. (Original) The article of manufacture of claim 22, wherein the script file and the document comprise logically separate data files.

28. (Original) The article of manufacture of claim 21, the method further comprising:

transmitting the transformed document to a client program.

29. (Previously Presented) The article of manufacture of claim 21, wherein the transforming step comprises:

retrieving a database value from a database; and

assigning the database value to an object of the DOM.

30. (Original) The article of manufacture of claim 21, wherein the transforming step comprises:

replacing a first object of the DOM with a different second object.

31. (Previously Presented) The method of claim 2, wherein the first object is an HTML file.

32. (Previously Presented) The system of claim 12, wherein the first object is an HTML file.

33. (Previously Presented) The article of manufacture of claim 22, wherein the first object is an HTML file.

34. (Previously Presented) The method of claim 2, wherein the transformation instruction is read from a script file located separately from the first object.

35. (Previously Presented) The system of claim 12, wherein the transformation instruction is read from a script file located separately from the first object.

36. (Previously Presented) The article of manufacture of claim 22, wherein the transformation instruction is read from a script file located separately from the first object.

37. (Previously Presented) The method of claim 2, wherein:

the first object is an HTML file;

the transformation instruction is read from a script file located separately from the HTML file; and

the HTML file and the script file contain information to indicate their correspondence to each other.

38. (Previously Presented) The system of claim 12, wherein:

the first object is an HTML file;

the transformation instruction is read from a script file located separately from the HTML file; and

the HTML file and the script file contain information to indicate their correspondence to each other.

39. (Previously Presented) The article of manufacture of claim 22, wherein:

the first object is an HTML file;

the transformation instruction is read from a script file located separately from the HTML file; and

the HTML file and the script file contain information to indicate their correspondence to each other.

40. (Previously Presented) The method of claim 1, wherein the document and the corresponding transformed document are in the same format.

41. (Previously Presented) The method of claim 40, wherein the same format is HTML.

42. (Previously Presented) The method of claim 1, wherein the value is changed in accordance with different users.

43. (Previously Presented) The method of claim 1, wherein the value is a variable.

44. (Previously Presented) The method of claim 1, wherein the value of the first object is empty before the first object is transformed.

45. (Previously Presented) The system of claim 11, wherein the document and the corresponding transformed document are in the same format.

46. (Previously Presented) The method of claim 45, wherein the same format is HTML.

47. (Previously Presented) The method of claim 11, wherein the value is changed in accordance with different users.

48. (Previously Presented) The method of claim 11, wherein the value is a variable.

49. (Previously Presented) The method of claim 11, wherein the value of the first object is empty before the first object is transformed.

50. (Previously Presented) The article of manufacture of claim 21, wherein the document and the corresponding transformed document are in the same format.

51. (Previously Presented) The article of manufacture of claim 50, wherein the same format is HTML.

52. (Previously Presented) The article of manufacture of claim 21, wherein the value is changed in accordance with different users.

53. (Previously Presented) The article of manufacture of claim 21, wherein the value is a variable.

54. (Previously Presented) The article of manufacture of claim 21, wherein the value of the first object is empty before the first object is transformed.

EVIDENCE APPENDIX

This Appendix is Not Applicable to the instant Appeal.

RELATED PROCEEDINGS APPENDIX

This Appendix is Not Applicable to the instant Appeal.